

oVirt

Live Storage Migration (Under The Hood)

Federico Simoncelli

Principal Software Engineer, Red Hat

February 2013

- Live Storage Migration in a Nutshell
- Live Storage Migration Types
- Some Prerequisites
- Constraints and Limitations
- General Overview
- Flow Overview
- SPM/HSM API
- Detailed Flow and Sequence Diagram
- Above and Below (oVirt Engine, libvirt and QEMU)

General Definition

- Live Storage Migration is the capability to move one or more VM disks from one storage to another without interrupting the guest operations

Motivations

- Facilitate storage hardware upgrades
- Move or clone VM disks across different (and eventually geographically separated) data centers

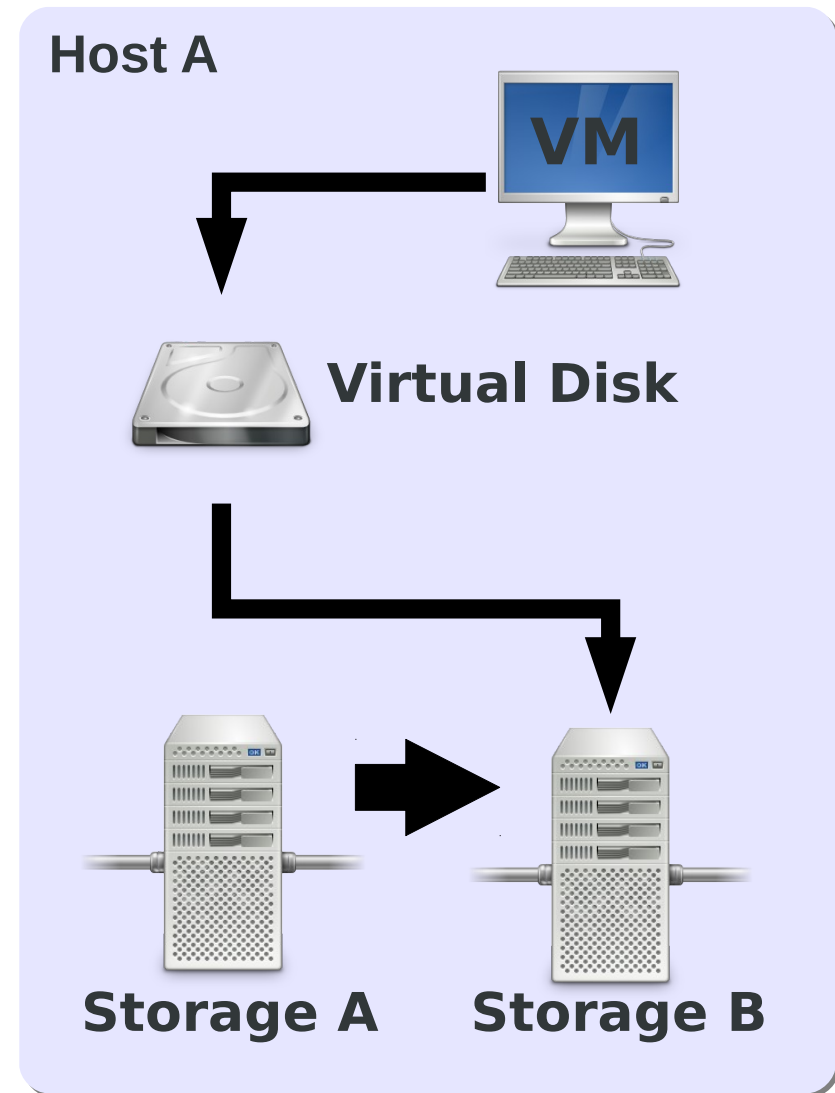
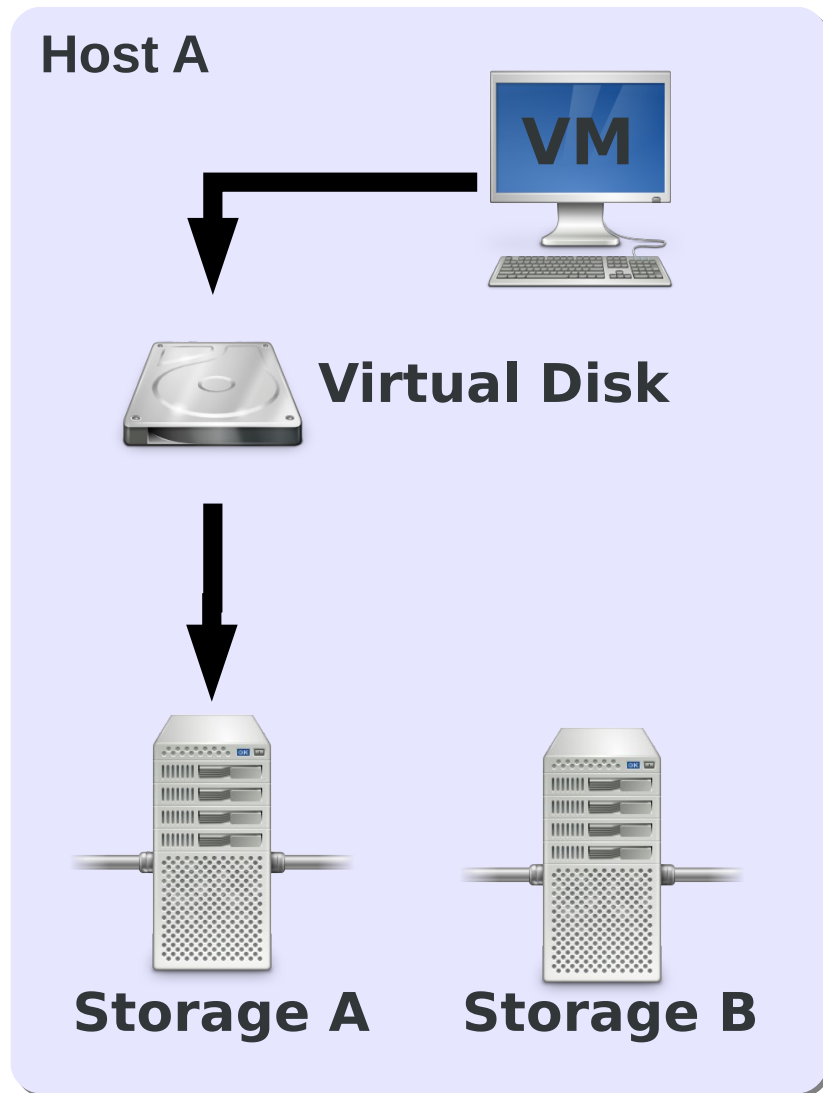
Without Shared Storage

- The involved hosts are not able to access both the source and destination storage
- The virtual machine is live migrated to a different host that is able to access the destination storage

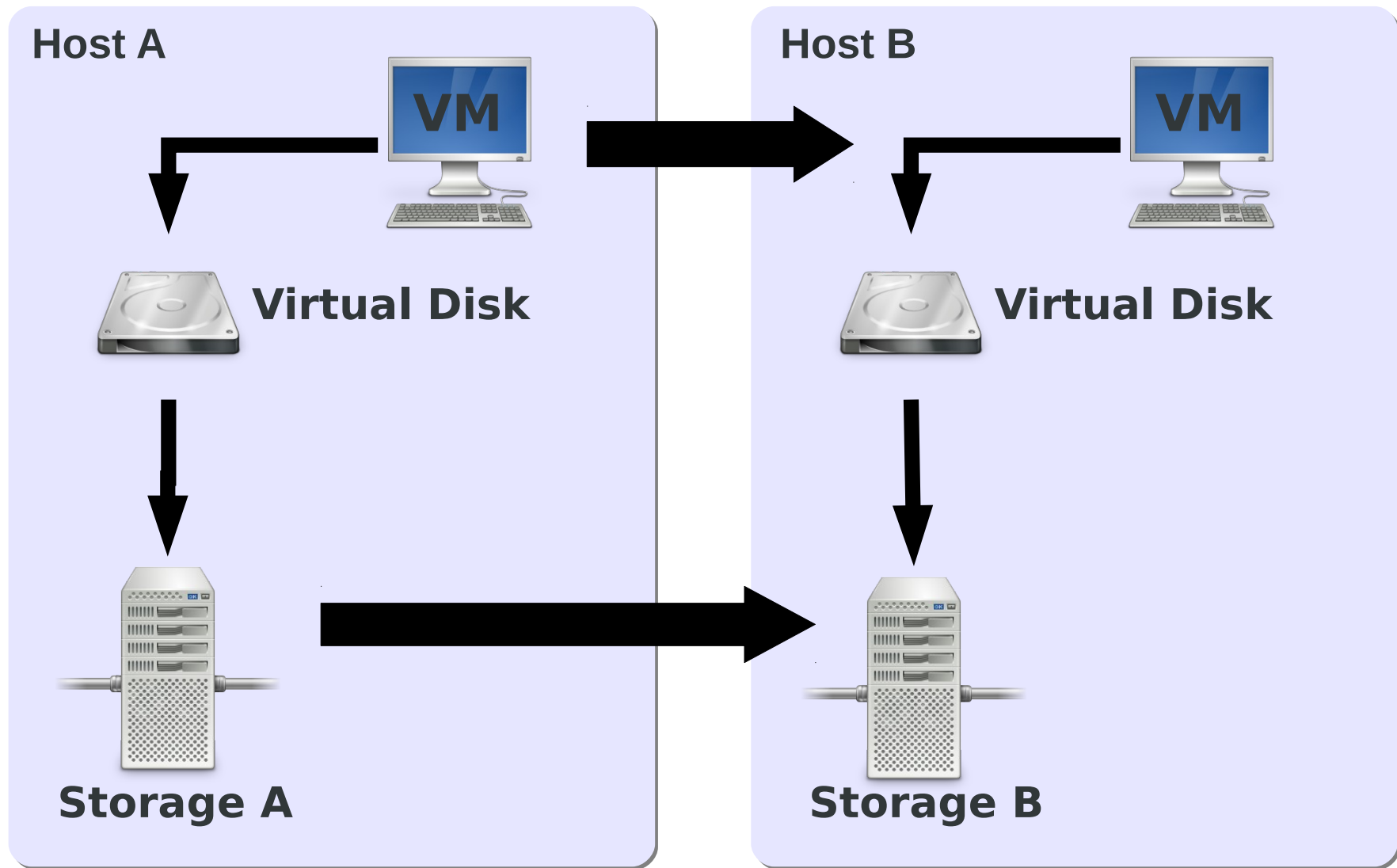
With Shared Storage

- The hosts involved are able to access both the source and destination storage
- The virtual machine is remaining on the same host

Overview With Shared Storage

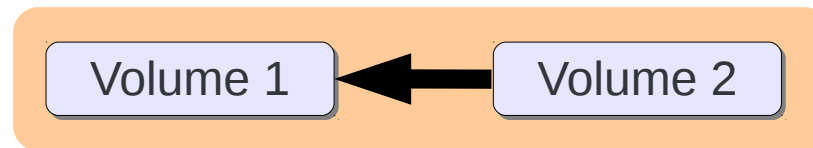


Overview Without Shared Storage



Some Prerequisites

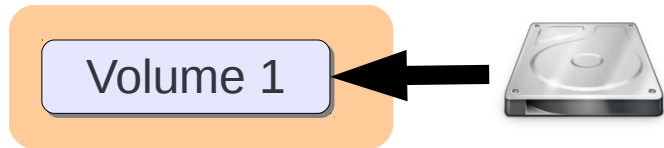
- General understanding of the oVirt architecture and few VDSM basics
- Virtual disks images as ordered collection (chain) of volumes, e.g.:



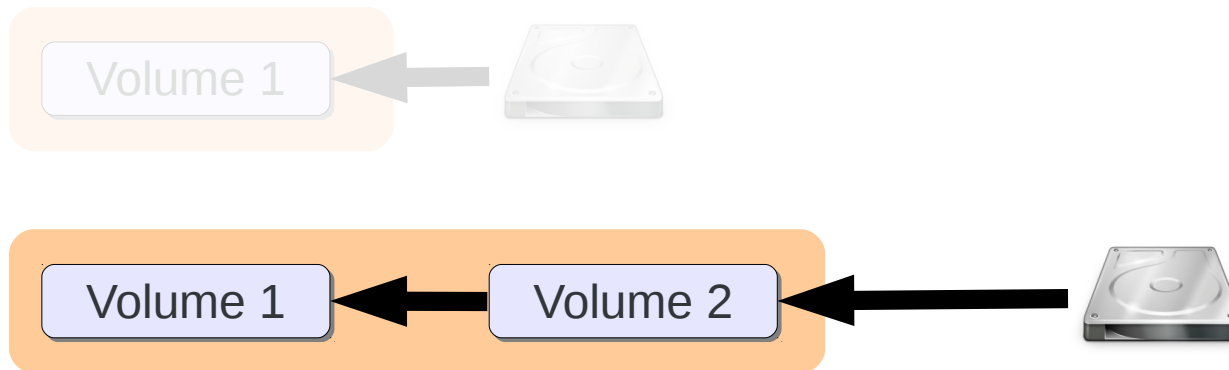
- General understanding of the QCOW format and how VDSM is handling the QCOW watermark on block storage domains

- All the image manipulations must be accomplished by the Storage Pool Manager (SPM)
- It is preferable to demand most of the data transfer to a specific entity, e.g.: any host in the cluster (SPM for simplicity), or a storage capability (eXtended Copy)
- Initial focus on live storage migration with shared storage and between homogeneous storage domain types
- An image (volume chain) should not be spread among multiple storage domains
- It is important to maintain the source image structure (“external snapshots”)

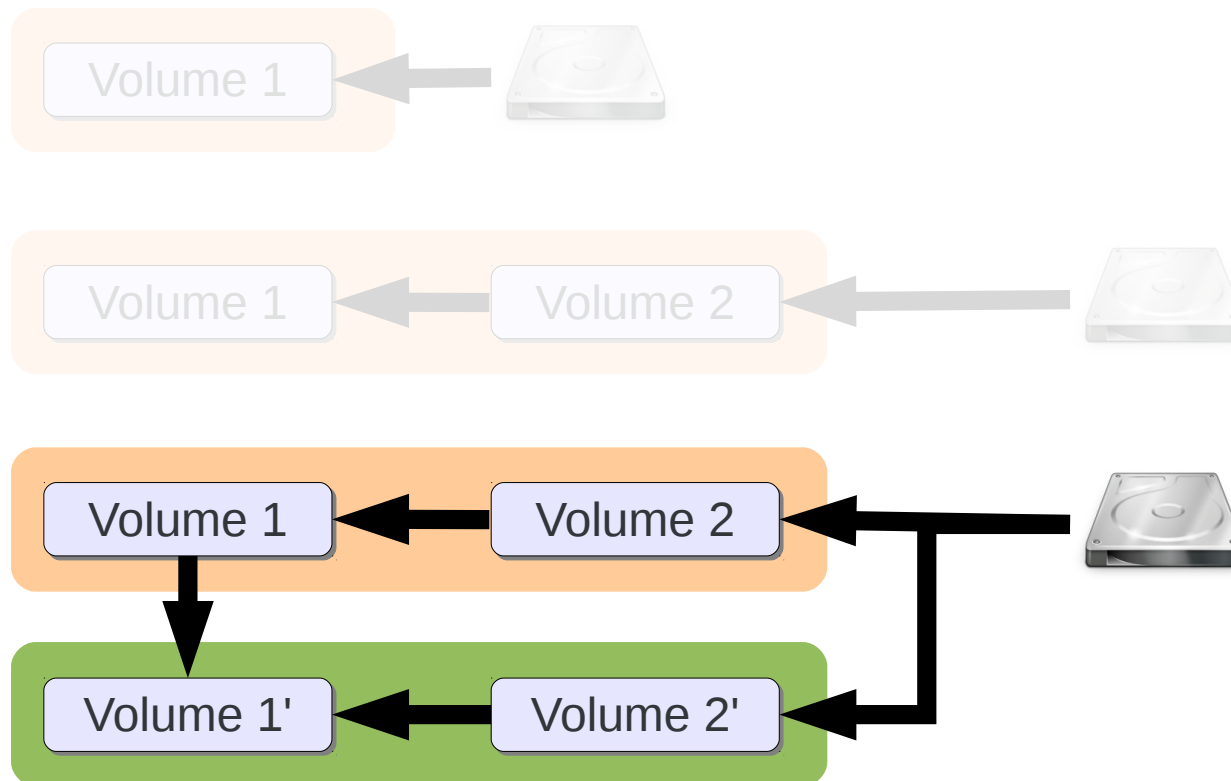
Flow Overview 1/4 – Initial State



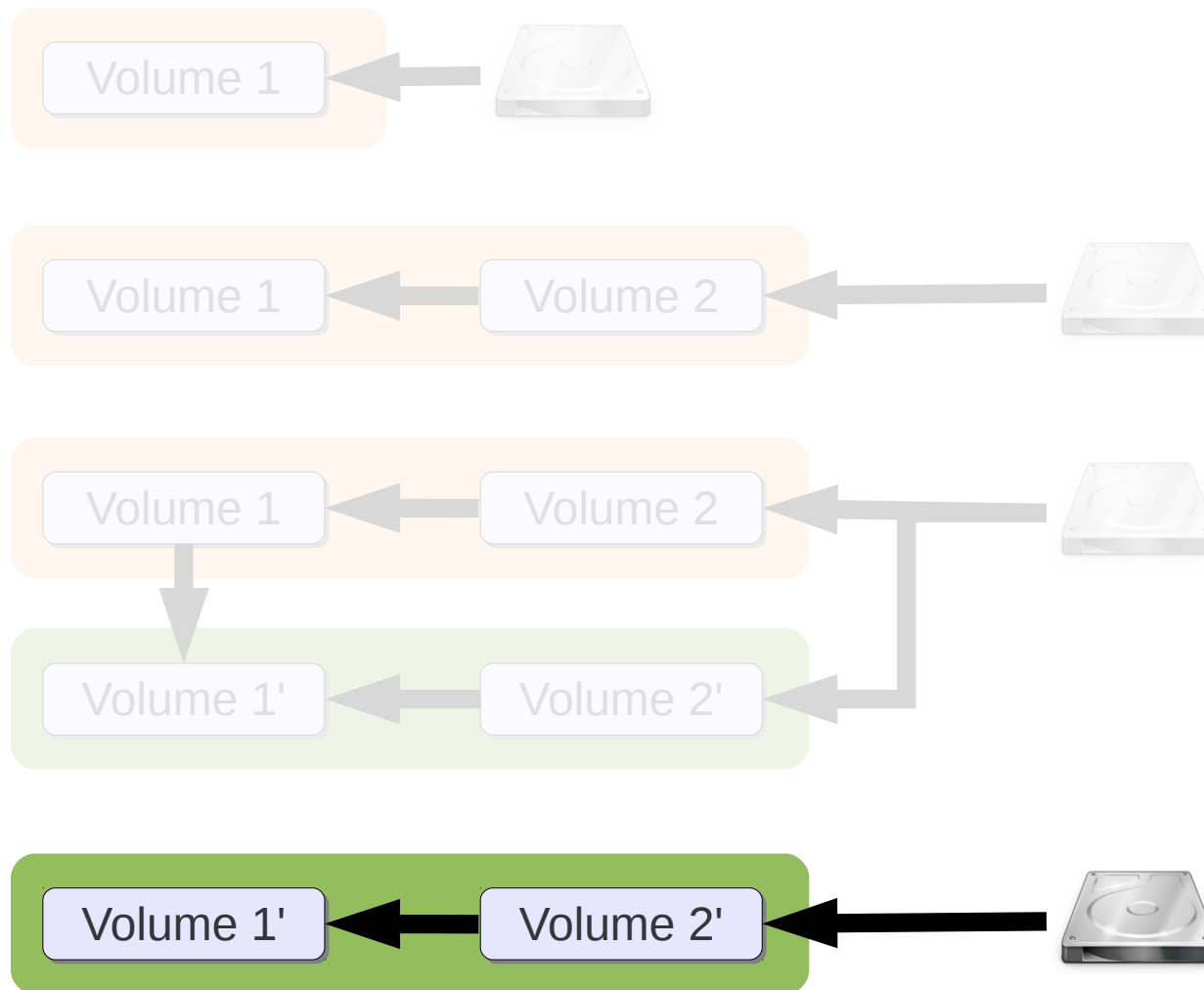
Flow Overview 2/4 – Live Snapshot



Flow Overview 3/4 – Replica & Copy



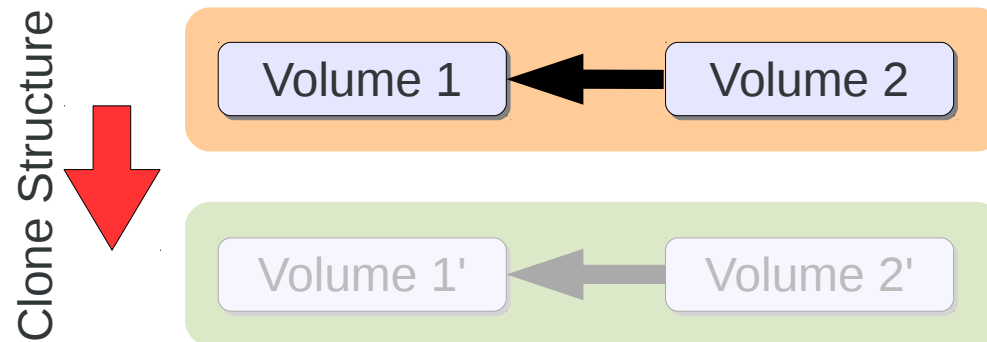
Flow Overview 4/4 - Completion



SPM API – cloneImageStructure

`taskId = cloneImageStructure(spUUID, sdUUID, imgUUID, dstSdUUID)`

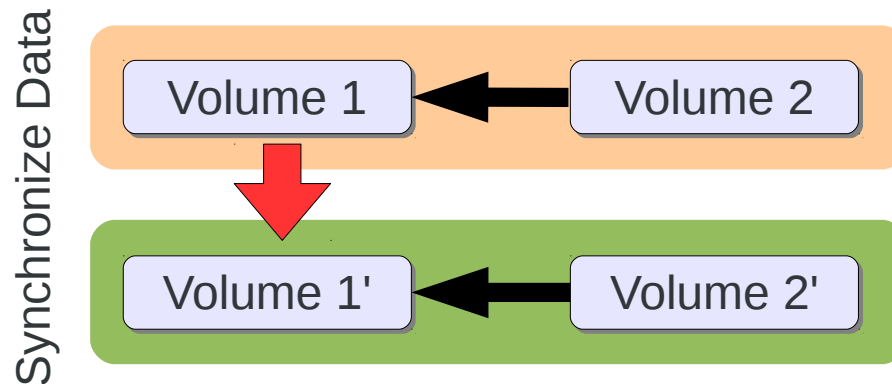
- **spUUID** storage pool
- **sdUUID** source storage domain
- **imgUUID** image to clone
- **dstSdUUID** destination storage domain



SPM API – syncImageData

`taskId = syncImageData(spUUID, sdUUID, imgUUID, dstSdUUID, syncType)`

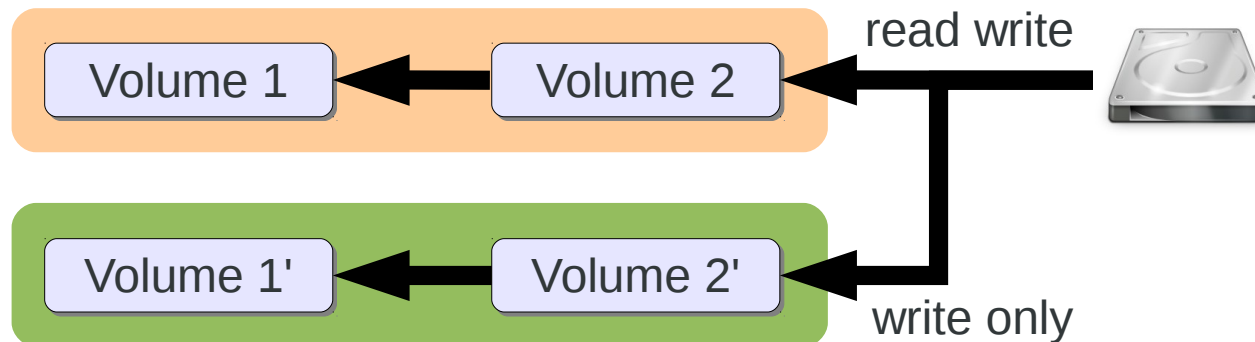
- **spUUID** storage pool
- **sdUUID** source storage domain
- **imgUUID** image to clone
- **dstSdUUID** destination storage domain
- **syncType** synchronization type (ALL, INTERNAL, ...)



HSM API – diskReplicateStart/Finish

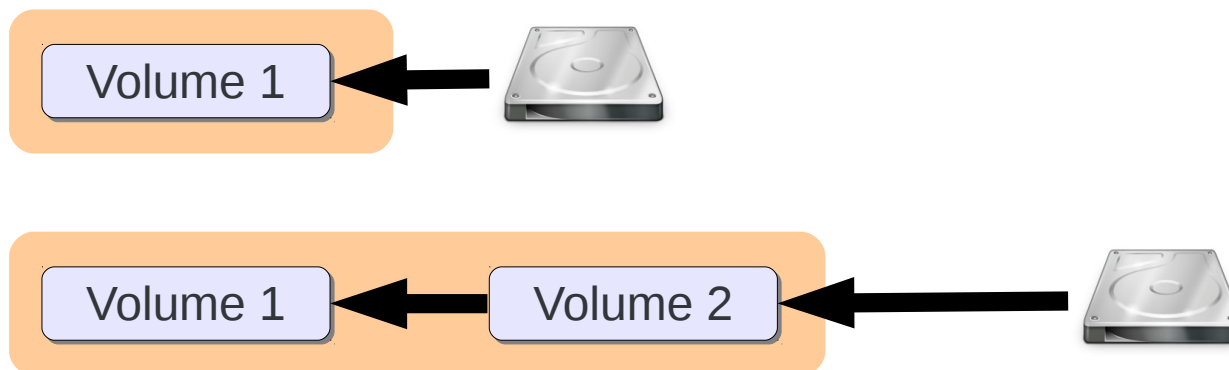
```
result = diskReplicateStart(vmId, srcDisk, dstDisk)
result = diskReplicateFinish(vmId, srcDisk, dstDisk)
```

- **vmId** virtual machine id
- **srcDisk** source disk
- **dstDisk** destination disk



Detailed Flow – Initial Live Snapshot

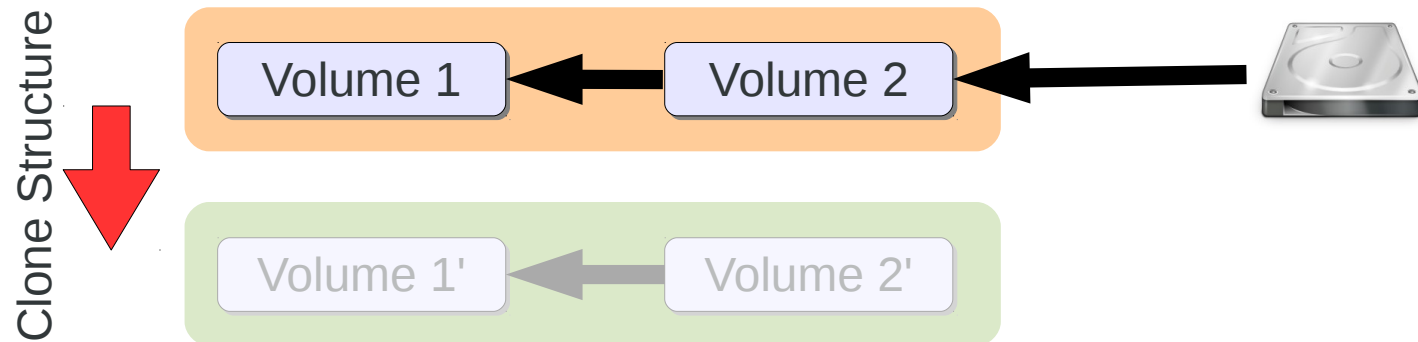
- SPM/HSM – initial Live Snapshot to minimize the amount of data replicated by the qemu process



Detailed Flow – Clone Image Structure

- SPM – clone the image structure from the source storage domain to the destination storage domain

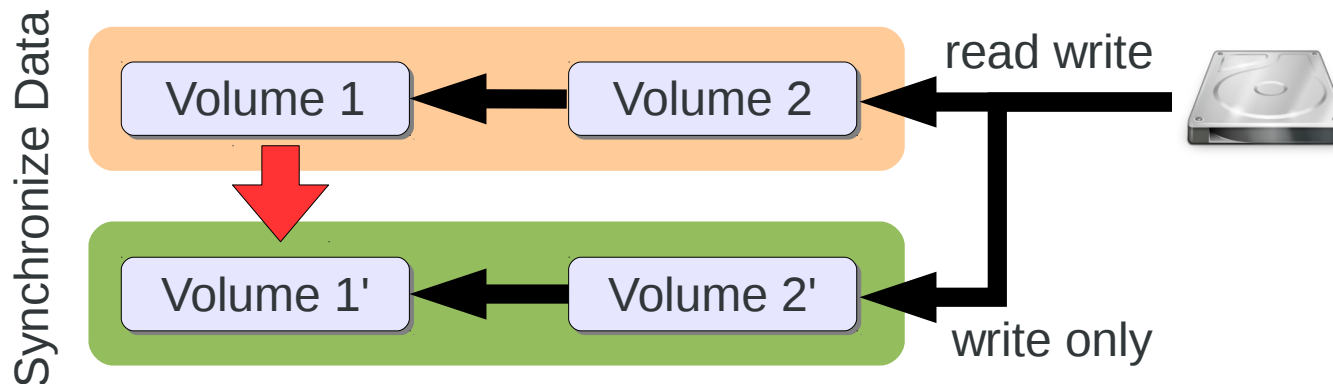
```
taskId = cloneImageStructure(spUUID, sdUUID, imgUUID, dstSdUUID)
```



Detailed Flow – Replicate and Sync

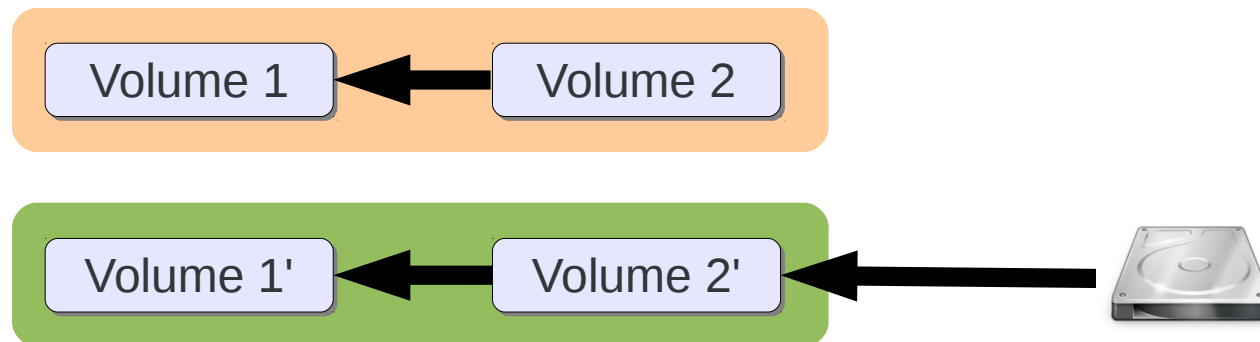
- HSM – start replicating the virtual machine writes on the destination storage domain
- SPM – synchronize the internal volumes data

```
result = diskReplicateStart(vmId, srcDisk, dstDisk)
taskId = syncImageData(spUUID, sdUUID, imgUUID, dstSdUUID, syncType)
```



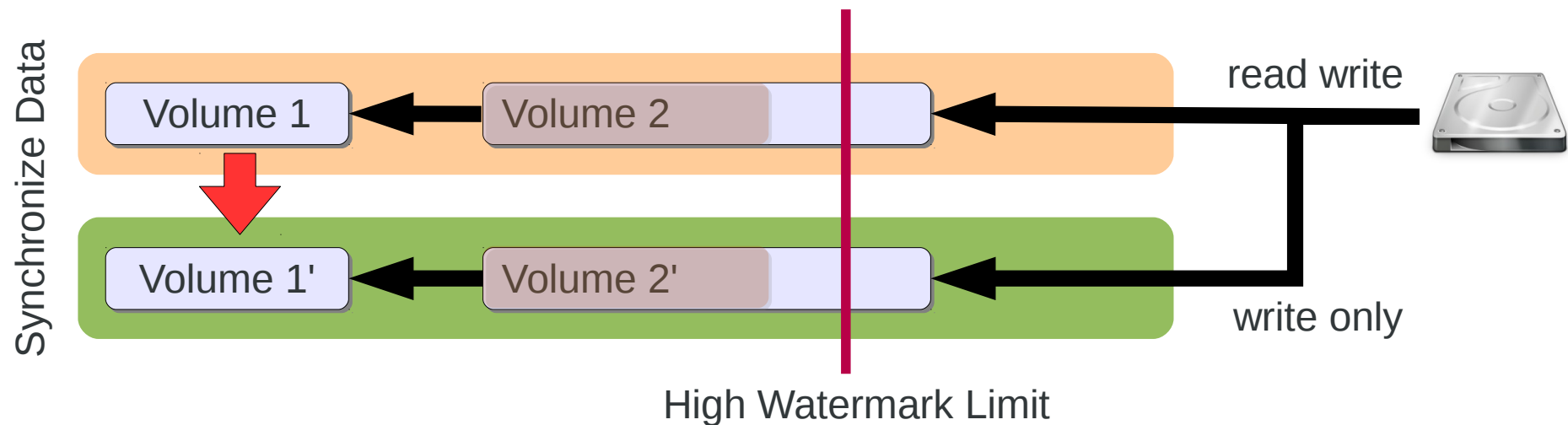
- HSM – complete the switch to the destination storage domain

`result = diskReplicateFinish(vmId, srcDisk, dstDisk)`



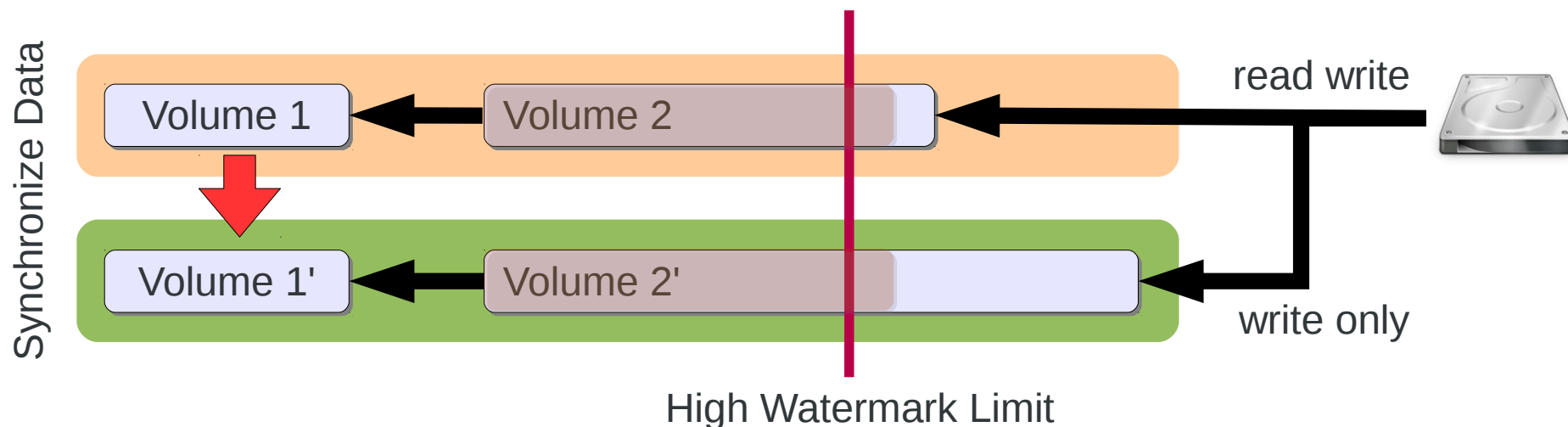
Detailed Flow – High Watermark 1/3

- The watermark limit is monitored on the VM host as for any other regular virtual disk



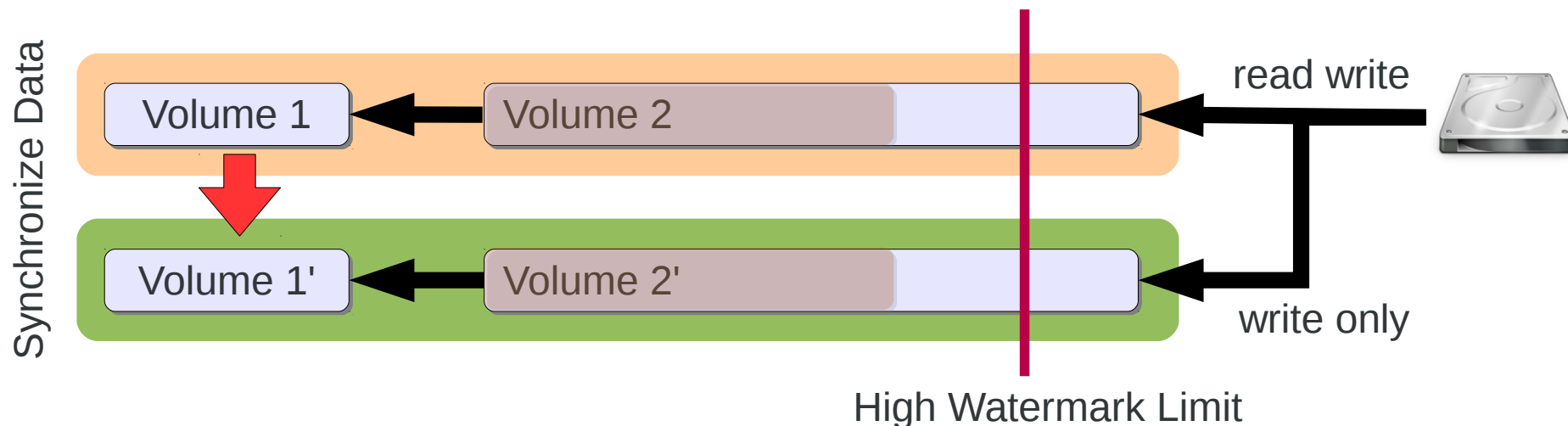
Detailed Flow – High Watermark 2/3

- The source and destination volumes data is written using the QCOW cluster size granularity (the size is the same)
- When the watermark limit is reached the destination volume is extended first



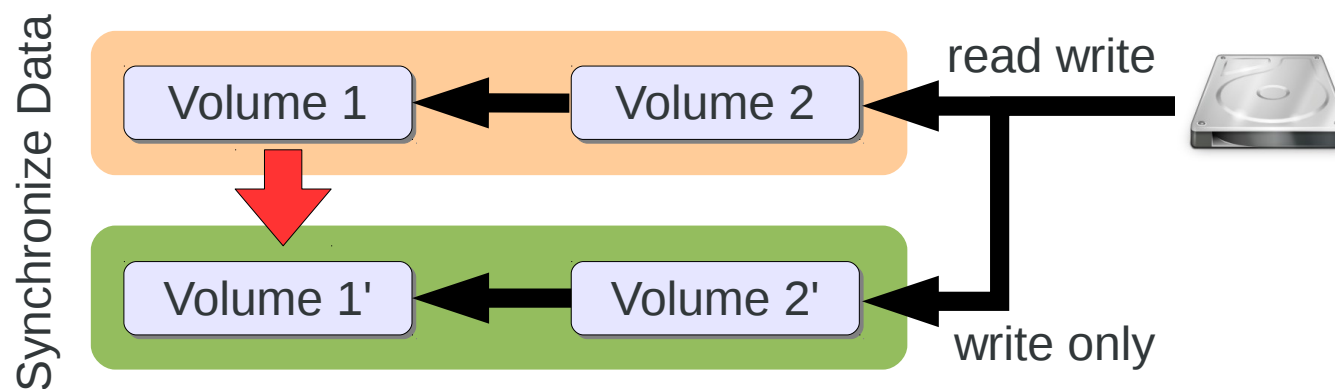
Detailed Flow – High Watermark 3/3

- After the destination volume has been successfully extended also the source is extended
- In case any of the extensions fails the VM is paused (ENOSPC) and the replica is not interrupted



Detailed Flow – Error Handling

- In case of errors it is possible to interrupt the replication and fallback to the source storage domain

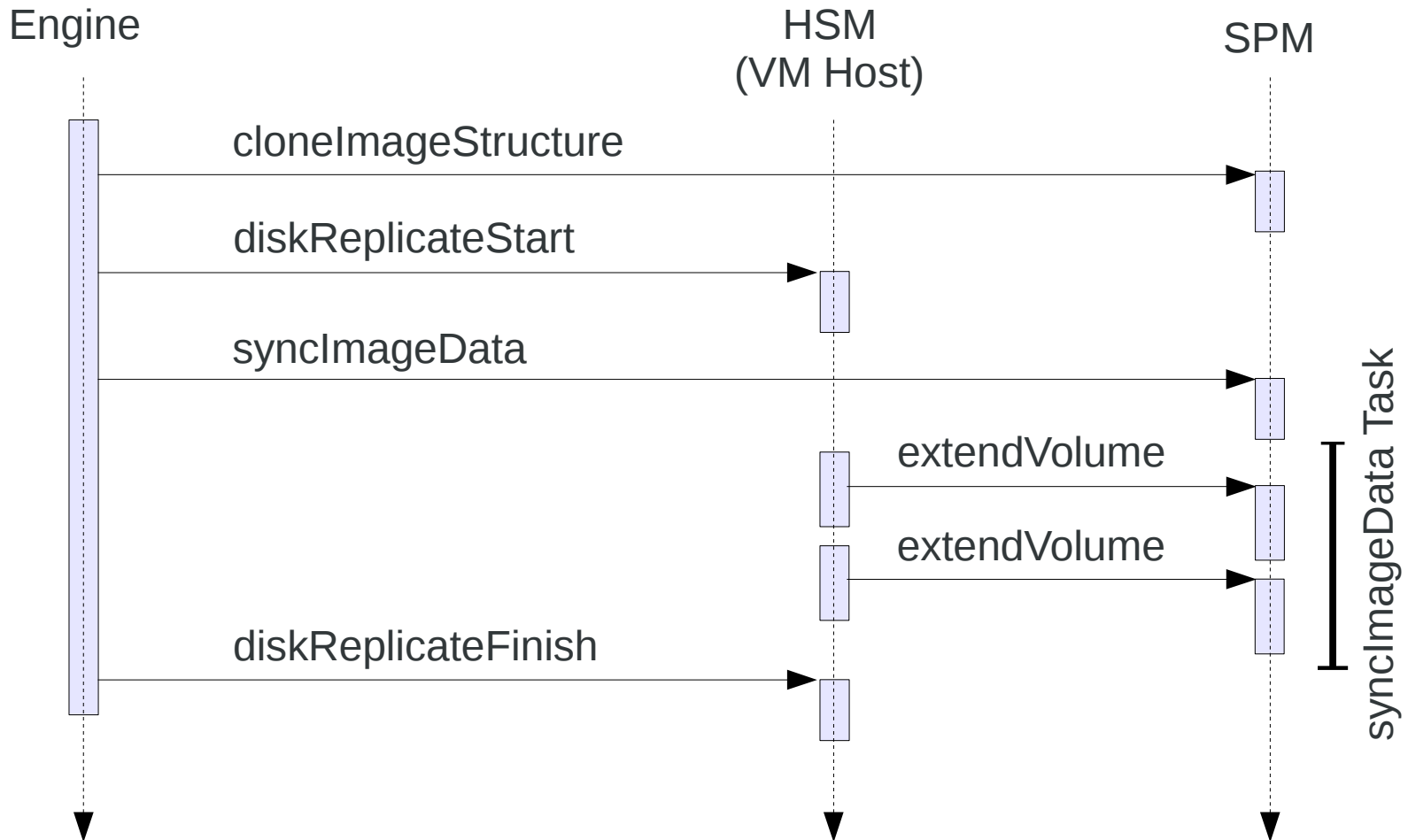


`result = diskReplicateFinish(vmId, srcDisk, srcDisk)`



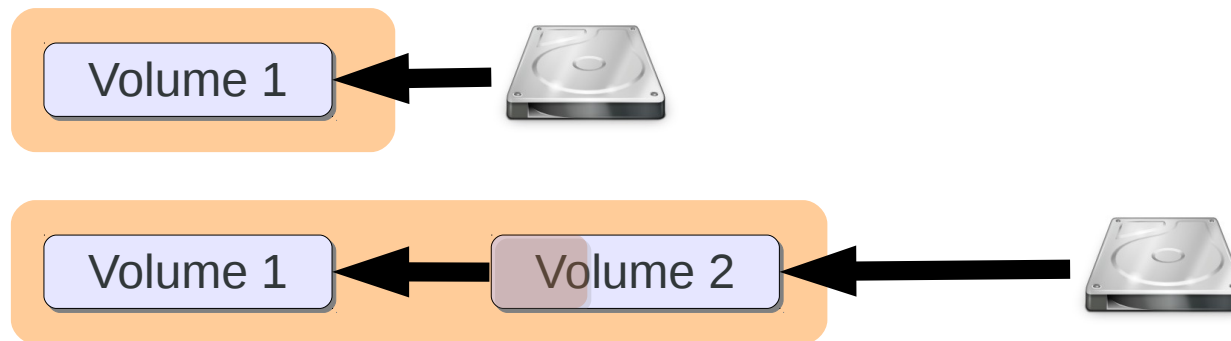
Sequence Diagram

Preliminary Live Snapshot

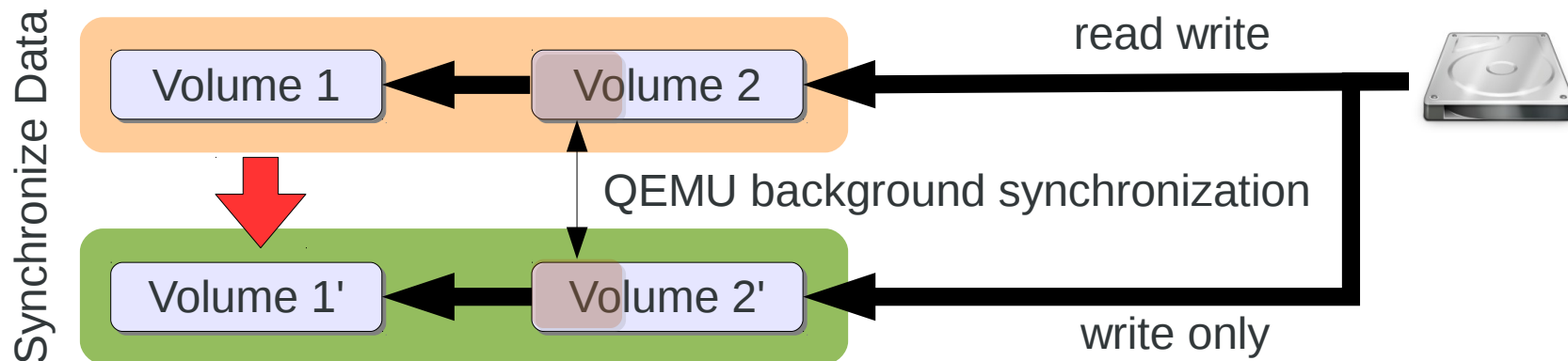


In Depth – Closing the Gap

- Live snapshot and disk replication are not one atomic operations



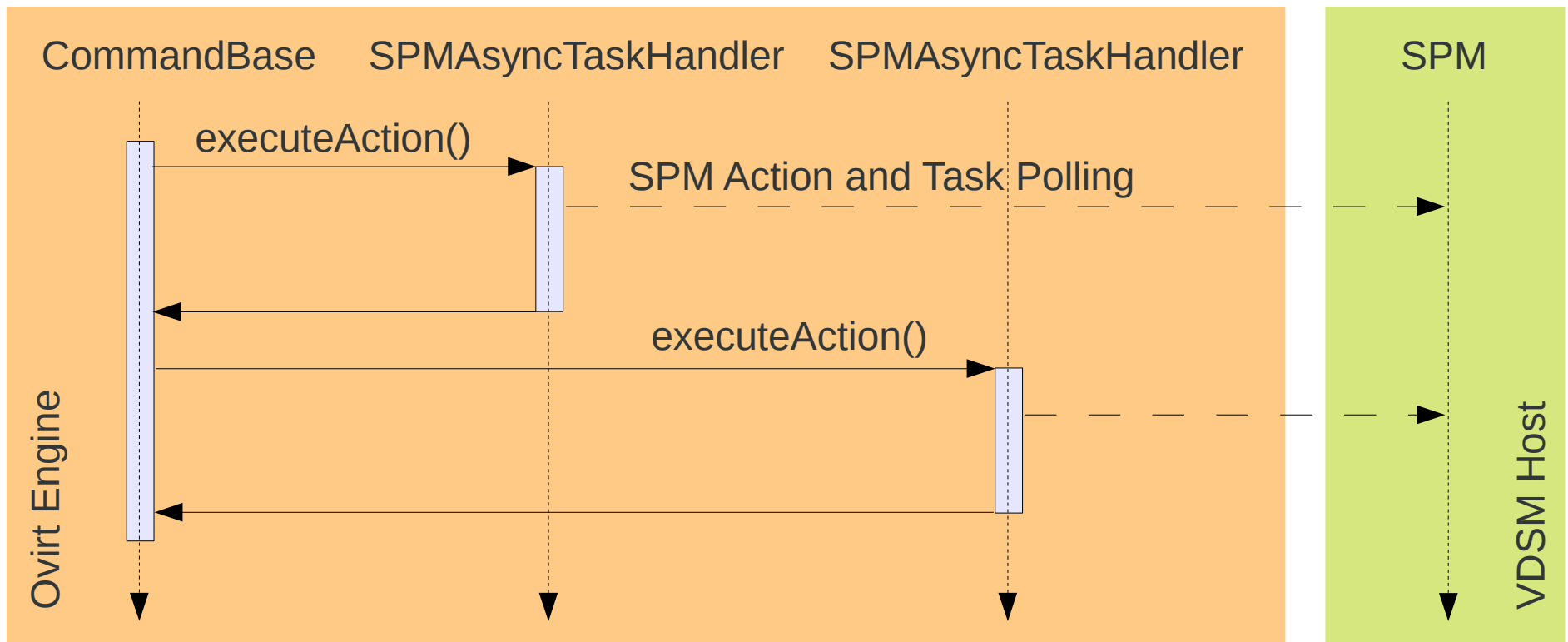
- A small amount of data is synchronized by a QEMU block job



- Relevant libvirt API:
 - `blockRebase(disk, base, bandwidth, flags)`
 - `VIR_DOMAIN_BLOCK_REBASE_COPY`
Starts a copy (replica/mirroring) instead of a regular block pull
 - `VIR_DOMAIN_BLOCK_REBASE_SHALLOW`
Consider only the top volume (leaf)
 - `VIR_DOMAIN_BLOCK_REBASE_REUSE_EXT`
Reuse an existing pre-initialized volume (prepared by the SPM)
 - `blockJobAbort(disk, flags)`
 - `VIR_DOMAIN_BLOCK_JOB_ABORT_PIVOT`
Switch to the destination volume (destination storage domain)
- Relevant QEMU commands:
 - `drive-mirror`
 - `block-job-complete`

- Serial Execution of Asynchronous Tasks

Allow an engine command to fire a series of asynchronous SPM tasks in order to allow complex flows (e.g. Live Storage Migration) to be implemented



http://wiki.ovirt.org/wiki/Features/Serial_Execution_of_Asynchronous_Tasks_Detailed_Design

- **Useful Links in the oVirt Wiki <http://wiki.ovirt.org/wiki/>**
 - /Features/Serial_Execution_of_Asynchronous_Tasks_Detailed_Design
 - /Features/Design/StorageLiveMigration
 - /Features/Serial_Execution_of_Asynchronous_Tasks
- **Mailing lists**
 - arch@ovirt.org users@ovirt.org
 - vdsm-devel@lists.fedorahosted.org
 - engine-devel@ovirt.org
- **IRC**
 - #ovirt on OFTC
 - #vdsm on Freenode